Abstract - Business process modeling and execution in SOA requires a set of methodologies and tools which support transition from analysis to execution level. Web services play a significant role in application development in SOA environment and publishing its functionality on registries to link their data and operations for different applications. Web service selection must satisfy not only the functional requirements but also the Non Functional Requirements (NFR) of the user. Based on our literature survey, we observe the need for improvement in current approaches so as to consider NFR during web service selection. Further, as number of users and their specific requirements increase, NFR conflicts are bound to rise and need to be understood. Detecting them and finding their impact on the system is the next rational step. Our work proposes to detect these conflicts using Ontology, Unified Modeling Language (UML) and UML Profile. The focus is on selecting an appropriate web service so that the Quality of Service (QoS) values are maintained. Our contribution is to develop a model driven approach so as to allow the designer to choose an appropriate web service. This is expected to greatly reduce the development and operational time besides providing transparency.

Keywords: Web Service Selection, QoS, Ontology, UML Profiles. Non Functional Requirements

1 Introduction

Web services based on ubiquitously adopted internet standards and supporting interoperability across different platforms have introduced a new era in application development. With an ever increasing number of web services providing similar functionality, Quality of Service (QoS) is becoming an important criterion for selection of the best available web service. QoS becomes a significant concern for service consumers and providers during service selection. Users need to know QoS information, reliability of the information and also the performance impact of a wrongly chosen web service. However, representing, storing and understanding the interdependency of the QoS values is an issue and needs due attention.

When discovering web services, clients look for those web services that meet their functional requirements. The service descriptions are manually scanned and those services which satisfy user requirements are selected or composed. With an increasing number of web services providing similar functionalities, the discovery process now emphasizes on how to find the service that best fits the consumer’s NFR.

NFR based service selection is not possible with the traditional mechanism as they do not consider user constraints. So the need of the hour is a system that assists the user to search and incorporate these capabilities that defines the QoS factors. QoS is defined as a set of quality requirements present in the collective behavior of one or more object parameters and are a set of non functional attributes like service response time, throughput, reliability and availability. They sometimes refers to the level of quality of service, i.e. the guaranteed service quality.

Our investigation raised certain issues:

i. What is an appropriate method of description of QoS values?

ii. After the filtering of web services what method is apt for non functional requirement based selection?

iii. How will the user know that the shortlisted web service will meet a distinctive non functional requirement?

iv. What is the best way to judge the extent to which QoS values of the user and provider agree?

v. How to associate the weights for each of the user’s non functional requirements?

vi. Do the NFR conflict and if they do to what extent do they affect the working of the system and its QoS value?

The goal of this research is to investigate how dynamic web service selection can be realized to satisfy a customer’s QoS requirements using a new model that can be accommodated within the existing web service selection methods. This paper focuses on study of the possibility of improving the proposed UML profile, including the OCL (Object Constraints Language) for the representation of user constraints during selection. This way the developers’ whose knowledge does not extend beyond UML can develop applications that use semantic web services. We propose a method to detect these conflicts using Ontology, Unified Modeling Language (UML) and UML Profile. Even though matching of QoS factors have been understood, it is more important to understand how NFR conflicts and how it can affect the system.

The remainder of this paper is structured as follows. We outline the motivation in section 2. Section 3 expresses use
of Model Driven development in web service selection. Section 4 describes the method of representing NFR in UML diagrams. Section 5 overviews verification of class diagram and OCL and Section 6 concludes the paper.

2 Motivation

The preface makes it evident that the first step for the support of QoS of web service is to provide a framework which considers the NFR of service provider and requester. The idea in the proposed design and implementation is to maintain all features with the standard web service selection mechanisms and expand them to support QoS characteristics. As a result, a web service user can pick among services the one that suits him/her most and further refine his options using quality criteria. To be widely adopted by users and to succeed in real-world applications, the development must catch up with mainstream software trends like Model Driven Development.

We can broadly classify web service selection methods into those discovered by functional requirements and those by non functional requirements as shown in Fig. 1.

![Fig. 1: Web Service Discovery Methods](image-url)

During design time or static web discovery that concentrates only on functional description, the application designer makes use of service registries and service descriptions to select and test binding to a service. As number of web services grows they share similar functionalities, but possess different non-functional requirements. The web service discovery and selection methods that use NFR have been investigated and a backdrop for motivation of our research is created. Solutions to service selection based on QoS problem can be roughly divided into three categories:

1. **Self Advertising**: Web service providers will describe their expected QoS information. A disadvantage is that provider may not be neutral in describing its own QoS information.

2. **Web service consumers experience about service quality**: QoS data are collected by other user's feedback or by active monitoring. A drawback of this approach is its complexity and overhead implementation.

3. **Third party evaluation of a web service owner**: It will test the web service and it’s published QoS information. This method is expensive and inflexible to implement.

Kokash et.al [2] suggested adding QoS values to UDDI by adding properties to property bag. Use of Quality Requirements Language (QRL) for QoS information was described by Ran et.al. [3]. However it does not provide sufficient information on when and how to control and manage any specified QoS information. The paper further suggested the use of a QoS certifier. The new certifier’s role is to verify service provider’s QoS claims. The proposed extension to the current UDDI design may not always be feasible and the verification of QoS properties at the time of registration may not guarantee up-to-date QoS information.

Maximillen et.al. [4] suggests use of agents based on distributed reputation metric models. Rajendran et.al. [5] proposed the Multi-Agent based architecture for both services registration and service discovery. The architecture utilizes the services of response agent, certification agent and query agent. D’Mello et.al [6] presents a repository to store, retrieve and act with QoS information. Efshani et.al. [7] uses a QoS broker to manage the interaction of QoS information between service requester and service provider and further effort is being placed on how to find the service that most coordination the consumers’ requirements. Al-Masri [8] proposes use of Web Service Relevancy Function used for measuring the relevancy ranking of a particular web service based on client’s preferences, and QoS metrics.

Another approach suggests use of WS-Policy [9]. WS-Policy is a specification that allows web services consumers and producers to advertise their policy requirements. Certain solutions suggest a conceptual model that is based on web service reputation and user feedbacks [10][11]. They however suffer from the fact that service do not provide guarantee as to the accuracy of QoS values over time or having up-to-date QoS information. QoS Ontology based solutions [4] uses a multiagent framework based on ontology for QoS. The ontology provides a basis for providers to advertise their offerings, for consumers to express their preferences, and for ratings of services to be gathered and shared. Baocai et.al. [12] proposes a framework that supports the automatic discovery of web services using QoS ontology. However they suffer from performance problems due to the use of ontology reasoners.

Where matching of web services is concerned, work by Baocai et.al [12] and Kritikos et.al [13] draws attention to semantic similarity, so as to improve the precision of service discovery. However the interface of web service has
difficulty to describe the service correctly. Moreover users also find it hard to exactly present their requirements.

In general, the widespread understanding and use of web services should be promoted to enable development of ubiquitous computing and for widespread adoption of web services. However, the learning curve for semantically rich languages can be steep. This fact provides a barrier to adoption and widespread use. So, development must catch up with mainstream software trends, as for example, the Model Driven Architecture.

We propose the use of UML profile and OCL constraints, to describe the NFR requirements and constraints of web service with the help of QoS Ontology. Note that the OCL is language familiar for the average software developer and such a selection process at design time could improve the overall performance by reducing conflicts at run time.

3. Model Driven Development for Web Service Selection

Model Driven Development (MDD) is a style of software development where the primary software artifacts are models from which code and other artifacts are generated. A model is a description of a system from a particular perspective, omitting irrelevant detail so that the characteristics of interest are seen more clearly. In MDD a model has to be machine-readable.

In MDD, models are used not just as sketches or blueprints but as primary artifacts from which efficient implementations are generated by the application of transformations. It has the potential to greatly reduce the cost of solution development and improve the consistency and quality of solutions. It does this by automating implementation patterns with transforms, which eliminates repetitive low-level development work. Rather than repeatedly applying technical expertise manually when building solution artifacts, the expertise is encoded directly in transformations. This has the advantage of both consistency and maintainability.

MDD shifts the emphasis of application development away from the platform allowing developers to design applications without concern of platform-level concepts. A software development project needs to produce many non-code artifacts and many of these are completely or partially derivable from models. The advantages of an MDD approach are as follows:

- Increased productivity
- Maintainability
- Reuse of legacy
- Adaptability
- Consistency
- Repeatability
- Improved stakeholder communication
- Improved design communication

When designing a solution, we must consider non-functional characteristics such as security and performance. In an MDD approach, it is often possible to capture many decisions related to non-functional characteristics in transformations. However, it is not always possible or desirable to completely automate these aspects of a solution. Solution-specific design may be necessary. In such cases, we introduce modeling techniques relevant to specifying non-functional characteristics. For example, we might introduce stereotypes that indicate the kind of traffic that is expected over a connection (frequent/infrequent, high volume/low volume). The transformations then use this information to generate implementation artifacts that are optimized for these performance characteristics.

4. Representing NFR Using UML Profiles

UML editors are ubiquitous in the software industry, and many can be updated to recognize new profiles. UML documentation of the requirements engineering process will sit more comfortably with all other UML documentation for a software project. A domain-oriented design approach that provides mechanisms for illustrating NFR using UML classes would facilitate the mapping from one domain to the other also.

Our work proposes to design a system which can assist in NFR based web service selection process in Service Oriented Architecture (SOA) environment. For illustration purpose, we have created a framework for UML profile corresponding to a Chronic Obstructive Pulmonary Disease (COPD) patient. This profile is meant to ease the work of software developers.

An important part of UML is the Object Constraint Language (OCL) – a textual language that allows to specify additional constraints on models in a more precise and concise way than it is possible to do with diagrams only. UML Profile contains stereotypes and tagged values. Stereotypes are attached to model elements to convey the meaning of those elements. Tagged values are name/value pairs. They are attached to model elements in order to supply additional information which is needed in the transformation process. We suggest the use of a precise approach that allows an analysis and validation of UML models and OCL constraints.

i. Case Study: COPD Patient

In e-health Remote Patient Monitoring (RPM) refers to variety of technologies designed to manage and monitor health conditions of a patient. RPM come with many
advantages like reduced cost, early intervention, integration of care and increased productivity. Chronic Obstructive Pulmonary Disease (COPD) is a slowly progressive disease of the airways characterized by a gradual loss of lung function. This leads to a limitation of the flow of air to and from the lungs, causing shortness of breath (dyspnea). The symptoms of COPD can range from chronic cough and sputum production to severe disabling shortness of breath.

These are the first signs of complications in the patient’s condition and the risk associated with it. Many COPD patients monitor their vital capacities and peak flow rates at home. This monitoring helps them and their physician to monitor the patient’s condition. The proposed model will be helpful to these COPD patients being monitored remotely to predict the onset of a spasm and alert the medical professional. Due to the prevalence and economic burden of hospitalized COPD patient, it is necessary to seek out methodologies that would facilitate the prevention, monitoring and treatment of them. Experts and researchers suggest monitoring and tracking patients’ symptoms on an everyday basis in order to prevent emergencies.

We consider this setup in a Service Oriented Environment (SOA) where web services will be used to transfer data securely. The selection of web services is critical as each patient’s NFR are unique. However they have to match the NFR offered by web service, and this is where our model comes into use. We try to select the most appropriate web service that delivers the highest QoS by matching the NFR or unique requirements of the user.

**ii. System Design**

Fig. 2 shows the data flow of the proposed system.

![Data Flow Diagram](Image)

The specifications of the present system are studied to determine what changes will be needed to incorporate the user needs. The input will consist of the user specifications, and the output will be the identity of the web service that can deliver highest QoS.

**iii. Non Functional Requirements Diagram**

NFR Diagram is used to understand how NFR are interrelated. For our discussion we consider two important NFR: security and response time. Later we will increase the scope of the NFR under consideration.

The NFR are different from the functional characteristics in the sense that they are not always completely satisfied. The set of NFR are interrelated and if they are conflicting, they affect the working of the system. To understand this, we draw NFR diagram, a visual representation of the decomposition of NFR. In this sense, the diagram shows how various NFR are rationalized. In analyzing NFR, one does not analyze them independent of each other, but rather in relation to each other. There can be looser relationships where one NFR considers, prevents or contributes towards the fulfillment of another.

**a. Security NFR Diagram:** The decomposition is shown in Fig. 3 below. When the patient’s vital data is sent, it is recognized as normal or abnormal condition. In case it is abnormal, the login protocols are bypassed and the vital data validation is done. This is done to avoid any delays to authentication and authorization.

![Security NFR Diagram](Image)

However in normal conditions, the patient’s data is validated through the Login Process. It consists of Access Rule validation, identification and data confidentiality. This figure shows that security NFR itself can be decomposed into several NFR.

**b. Response Time NFR Diagram:** Response Time is defined as the elapsed time between the end of an
inquiry on a computer system and the beginning of a response. The total response time NFR consists of response time for validating data, logging in, response time for alert and then saving data.

All of the goals considered can be termed as soft goals, that is, they are yet too vague to be formalized. Since they are not sufficiently defined, it is also not yet clear what it would take to satisfy one of these goals. Ordinarily, a “reduces” relationship holds between a goal and a subgoal if satisfaction of the subgoal is sufficient for satisfaction of the goal it reduces. However, where a reduces relationship exists between goals in which any of the reduced goals are soft, we talk in terms of partially satisfying rather than satisfying.

Ordinarily, a “reduces” relationship holds between a goal and a subgoal if satisfaction of the subgoal is sufficient for satisfaction of the goal it reduces. However, where a reduces relationship exists between goals in which any of the reduced goals are soft, we talk in terms of partially satisfying rather than satisfying.

5. Verification of Class Diagrams and OCL

The idea of verification of web service suitability using MDD in the case of a COPD patient seems to be a promising idea. As a result it will be able to detect mismatch or a reduction in QoS values due to NFR conflicts. The approaches revolve around satisfiability property of a model, i.e. deciding whether it is possible to create a well-formed instantiation of the model.

A general outline for this work would be:

i. Depict the candidate web services in the form of a class diagram, making use of the WSDL file structure and its tags.

ii. Understand the input notation/requirement of the user. These are the OCL constraints. If an internal formal notation is used, it should be transparent to the designer.
iii. Analyze the effect of the OCL constraints on the class diagrams and how they may disagree with the design without any type of manual annotation.
iv. Provide results in a format meaningful to the designer.
v. Integrate seamlessly into the designer tool chain.

The expressivity of class diagrams is limited to class level interaction and constraints. The Object Constraint Language OCL is intended to extend a UML model (mainly class diagrams) with symbolic constraints[14]. There have been some methods that discuss the idea of UML/OCL constraint. However its use in the field of web service selection is innovative.

Cabot et al. [15] describe a CSP-based tool for reasoning about finite satisfiability of class diagrams that are extended with OCL constraints. UMLtoCSP is a tool for the automatic verification of UML models annotated with OCL constraints. It can check automatically several correctness properties about the model, such as the satisfiability of the model or the lack of contradictory constraints. Currently, the tool works on UML class diagrams only. UML2Alloy plays an important role to create a bridge between UML and Alloy.[16]. The tool takes an ArgoUML-generated XMI file in order to transform the UML model into Alloy code. It transforms the input into assertions, simulations, or invariants.

Nevertheless, there are a few obstacles that may prevent the introduction of WS provision. One of them is the inability to represent the non-functional features of WSs, i.e. their quality-of-service. To take care of this we second the use the formulation of a QoS ontology framework that is used to support QoS-aware web service selection [14]. A set of rules defined by Semantic Web Rule Language (SWRL) can also be described and designed for reasoning to acquire more advanced knowledge based on simple ones.

6. Conclusion and Future Work

This work elucidates the relevance of QoS during web service selection for a COPD patient remotely monitored. Remote monitoring involves critical data handling and data security keeping in mind the patients unique requirements. The report discusses the relevance of NFR during web service selection and the effect of conflicting NFR. The paper proposes use of Model Driven Development (MDD), UML Profiles and its extensions for ranking the most suitable web service from a list of functionally satisfying web services that are shortlisted. The benefits of MDD ranges from ease of understanding to its use during design stage to avoid conflicts during run time. The web service class diagrams and OCL constraints can be verified using a UML verification tool in future.

The proposed method is flexible and transparent and can be used along with any web service selection method. It also contributes to the improvement of service selection efficiency when service is retrieved in an automatic way. The major advantage of this approach is to decrease the complexity of web service selection to user as it can be included during design stage.

Our current interest as described in the paper focuses mainly in two directions:

i. Investigating the NFR conflicts and
ii. Understanding the effect of these conflicts on the QoS values.

In future, we would want to extend our experimental setup with additional scenarios. We also plan to extend QoS parameters to include information such as availability, reliability etc. Moreover, this approach may be extended to automatic service selection using multi-dimensional QoS.

7. References


Web Service Policy, Available at: http://www.w3.org/Submission/WS-Policy/


